Reliability & Performance Analysis of Multi-State Systems Based on Analytical Load Flow Considerations

Hindolo George-Williams
Institute for Risk & Uncertainty Engineering
University of Liverpool, UK

Institute of Nuclear Engineering & Science
National Tsing Hua University, Taiwan

Edoardo Patelli
Institute for Risk & Uncertainty Engineering
University of Liverpool, UK

Min Lee
Institute of Nuclear Engineering & Science
National Tsing Hua University, Taiwan

ABSTRACT: The last three decades have been marked by the advent of various analytical and simulation algorithms, enhanced for the reliability evaluation of multi-state systems. Though the latter are widely believed to be the most applicable to realistic systems, they impose a greater degree of computational burden. Consequently, they have been outshone, especially in structural optimization, redundancy allocation and maintenance optimization problems. On the flip side, analytical techniques are constrained by their various unique limitations. Prominent amongst these being, inapplicability to multiple output systems with competing demand and reliance on the enumeration of system path or cut sets prior to analysis. The development, therefore, of a single approach that addresses these limitations is desirable. In this paper, the fact that most engineering systems satisfy the flow conservation principle and can be regarded as multi-state flow networks is exploited. An analytical algorithm that efficiently derives all the possible system performance levels and uses basic probability algebra to estimate their probabilities of occurrence is developed. The algorithm is enhanced to support systems with flow losses, Common-Cause Failures (CCF), and minimal system reconfigurations. These attributes, as applied to two case studies, ensure the limitations of existing techniques are overcome.

1 INTRODUCTION

A multi-state system (MSS) is one in which the components, as well as, the system, can exist in more than two output levels. It occurs in various practical applications like power systems, transportation networks, communication systems, water distribution networks; to name but a few (Yeh 2015). For such a system, reliability and performance evaluation is effort intensive. Traditional binary-state system reliability evaluation techniques are often inapplicable without careful modification. In spite of these, numerous techniques have been devised for their reliability modelling. These, according to (Levitin 2004) belong to one of four categories; Monte Carlo simulation (George-Williams & Patelli 2016), modification of binary-state techniques (Zang, Wang, Sun, & Trivedi 2003, Yeh 2015, Yeh 2008, Lin 2002), stochastic process (Lisnianski, Frenkel, & Ding 2010) and the Universal Generating Function (UGF) (Levitin 2005). All four categories and by extension their derivatives, possess specific strengths and limitations, some unique only to them. A detailed review on these and the recent advances in MSS reliability are contained in (George-Williams & Patelli 2016, Lisnianski, Frenkel, & Ding 2010). In a broader sense, every MSS technique can be classed as either analytical or simulation-based. Graph-based algorithms and the UGF technique are arguably the most widely used analytical approaches to MSS reliability evaluation. They have, therefore, been singled out as the ref-

erence against which the approach proposed in this work is compared.

Graph-based algorithms are mainly based on the concepts of *minimal paths* or *minimal cuts* and normally use the *sum-of-disjoint-products* to obtain the reliability of an MSS (Yeh 2015, Yeh 2008, Lin 2002). However, all but very few (for example (Yeh 2008)) require derivation of the system's path or cut sets prior to analysis, which can be difficult for complex systems. They, cannot model systems with flow losses, impose that component capacities and system demand be integer valued and are yet to be applied to multiple output systems with competing demand (George-Williams & Patelli 2016). System performance is defined with respect to only one state at a time, obtaining the overall MSS performance, therefore, would require calling an algorithm for all system states. This may be time consuming for some systems.

The UGF, on the other hand, is able to algebraically obtain the overall MSS performance from the performance of its components. It's applicable even to systems with dependent components and systems prone to common-cause failures (Levitin 2005, Levitin 2004). However, like graph-based algorithms, it is inapplicable to systems with flow losses, as well as, systems with multiple competing demand. Also, though easily applicable to series-parallel systems, it's not intuitive for systems of complex architecture.

Most practical systems have complex architecture, and they, with their components, may have non-integer valued capacities. They may have multiple sources and outputs with competing demand and under some failure conditions, all or a fraction of system flow may be lost. For these systems, the approaches under review are inadequate, and a credible alternative is therefore required. This work proposes an alternative that possesses all the desirable attributes of these approaches but lacks their limitations.

The remaining sections are organised as follows; Section 2 gives an overview of the proposed approach. In Section 3, are details on how the system and its components are modelled. The algorithms and procedure to determine the most common reliability indices are presented in Section 4. Section 5 contains a series of case studies demonstrating the applicability of the approach. It's concluded with a sub section dedicated to general comments on the approach, its challenges and future. The conclusion is presented in Section 6.

2 PROPOSED APPROACH

The approach combines load flow principles and network theory to derive system performance levels. The system is represented by a graph in which components and demand points are nodes connected by edges. Its structure is defined by a square matrix, $\mathbf{A}$, known as an adjacency matrix. This matrix can be manipulated to provide the incidence matrix, $\mathbf{\Gamma}$, from which, two other matrices defining the flow across the system are derived. Given the capacity vector (vector containing capacities of nodes) of the system, the actual flow across every edge and node can be derived using the interior point algorithm (Mehrotra 1992, Kojima, Mizuno, & Yoshise 1989).

The overall performance analysis of the system proceeds by collating all the possible combinations of component states and their associated probabilities of occurrence. For each combination, the performance levels of output nodes are determined and recorded. From these, the set of unique performance levels for every output node is derived. Since each combination of component states (state vector) results in only one possible performance level, probabilities corresponding to each unique performance level are summed to obtain its probability of occurrence.

The flow equations are such that the effects of losses across nodes and edges and demand at output nodes are incorporated. As a result, the approach can model systems with flow losses and multiple demand quite easily. If every state vector is assigned an index, the set of indices corresponding to each system performance level can be deduced. These indices represent the indices of probabilities summed to obtain the probability of occurrence of the performance level in question. With the structure of the system and performance levels of nodes fixed, these indices, together with the sets of output performance levels, define the *performance signature* of the system. This performance signature enhances the study of system response to changes in the state probabilities of its nodes, without the need for repeated system analysis.

2.1 *Assumptions*

The following are the underlying assumptions of the proposed approach and its associated algorithms.

1. Flow through a node or an edge cannot exceed its capacity and all edges are perfectly reliable. Unreliable edges are treated as nodes.

2. Inflow to an intermediate node or an edge is equal to outflow plus any losses.

3. The system is free of nested CCFs. That is, Common-Cause Groups (CCG) are independent.

4. On arrival of a CCF event, the probability of failure of all nodes belonging to the particular CCG is 1.

3 MODELLING THE SYSTEM AND ITS COMPONENTS

Let E_i represent the properties of node i of the system, such that $E_i = (\mathbf{C}, \mathbf{S}, \mathbf{P}, \mathbf{D}, \Lambda)$. $\mathbf{C} = \{c_x\}^n \mid 0 \leq c_x \leq c_{max}$ is the node's capacity vector and specifies its performance (c_x) in each state (x), c_{max} is its maximum performance level and n its total number of

states. $\mathbf{P} = \{p_x\}^n$, is a vector defining the probability of each performance level in $\mathbf{C}$.

In some failure modes, outflow from the node may be less than inflow. The difference is dissipated in the node and it's lost. Examples of these are lossy power lines and a broken pipeline. If $\varepsilon_x \mid 0 \leq \varepsilon_x \leq 1$, is the fraction of inflow dissipated in the node when in state x, then $\mathbf{S} = \{\varepsilon_x\}^n$, defines the loss for all its node.

Another phenomenon applicable to components of some systems is minimum load restrictions. Sources and intermediate nodes are sometimes restricted from operating below a certain load level. When their effective load drops below this threshold, they are shut down, sometimes for reliability and/or cost considerations. To account for this, $\Lambda_i \mid 0 \leq \Lambda_i \leq c_{max}$ is introduced to define the threshold load level of node i.

CCF, the failure of a component or a group of components due to the same event may exist in the system. The Common-Cause Initiator (CCI) may be another component or may reside outside the system boundary. Examples of the latter are extreme environmental events, terrorist threats, human error and other similar events that may affect the operation of the system. A CCG arises if a group of components is linked to some CCI either by virtue of similarity in design, proximity, dependence on shared resources e.t.c (Levitin 2005). Appreciating that in the most general case, a CCI may induce in its CCG any state transition and not necessarily complete failures, matrix $\mathbf{D}$, as presented by Equation 1 is introduced to define the CCF events initiated by node i. Where $(d_{j1}, d_{j2}, d_{j3}, d_{j4}) \in \mathbb{Z}^+$

$$\mathbf{D} = \{d_{j1}, d_{j2}, d_{j3}, d_{j4}\}_{u\times 4} \mid j = 1, 2, ..., u-1, u \quad (1)$$

and d_{j1}; the state of i triggering the CCF, d_{j2}; the affected node, d_{j3}; the state the component has to be in to be affected (vulnerable state) and d_{j4}; its target state on occurrence of the CCF event. Each row of $\mathbf{D}$ therefore defines the behaviour of an affected node and u is the sum of affected nodes over all CCF events initiated by i. Therefore, $\mathbf{D} = \emptyset$ if node i is not a CCI.

3.1 *The System Model*

Let the components of the system, including demand points be numbered consecutively from 1 to M with the efficiency and capacity of the link between nodes i and j given by $\alpha_{ij} \mid 0 < \alpha_{ij} \leq 1$ and $l_{ij} \mid 0 < l_{ij} \leq \infty$ respectively. The system can be defined by $\mathbf{A}$ and the link capacity matrix; $\mathbf{L} = \{l_{ij}\}_{M\times M}$ such that each non-zero entry in $\mathbf{L}$ corresponds to a non-zero entry in $\mathbf{A}$. The properties of the latter are defined in Equation 2.

$$\mathbf{A} = \{a_{ij}\}_{M\times M} \mid a_{ij} = \begin{cases} \alpha_{ij} & \text{If flow is } i \rightarrow j \\ 0 & \text{Otherwise} \end{cases} \quad (2)$$

The edges of the graph are defined by a k by 2 matrix; $\mathbf{e}$, where, k; the number of edges is equal to the total number of non-zero elements in $\mathbf{A}$. Edge e_{ij} depicts the edge originating from node i and terminating on node j. $\mathbf{e}$ is obtained by traversing $\mathbf{A}$ from the upper left to the lower right element, exploring each column from top to bottom and extracting the i and j for each non-zero entry. Its properties are outlined in Equation 3, where $\mathbf{V} = \{1, 2, ..., M\}$ is the set of nodes.

$$\mathbf{e} = \{i, j\}_{k\times 2} \mid k = \sum_{j=1}^{M}\sum_{i=1}^{M}(a_{ij} > 0) \quad \forall (i,j) \in \mathbf{V} \quad (3)$$

All three properties of the graph can be defined by a single parameter $G \mid G = (\mathbf{V}, \mathbf{A}, \mathbf{L})$. If $\mathbb{E}$ is the set containing the properties, E_i of each node of the system, then the system structure and property can be defined by the set $\mathbb{S}$ as in Equation 4.

$$\mathbb{S} = (G, \mathbb{E}) \mid \mathbb{E} = \{E_i\}^M \quad \forall i \in \mathbf{V} \quad (4)$$

The incidence matrix; $\mathbf{\Gamma}$ defines the relationship between nodes and edges and it's related to $\mathbf{A}$ by Equation 5. The variable, $q = 1, 2, ..., k$ (the edge number) is the index of edge e_{ij} in $\mathbf{e}$ and $p = 1, 2, ..., M$.

$$\mathbf{\Gamma} = \{\gamma_{pq}\}_{M\times k} \mid \gamma_{pq} = \begin{cases} 1, & p = i \\ -a_{ij}, & p = j \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

$$\forall (i,j) \in \mathbf{e}$$

$\mathbf{\Gamma}$ is obtained by looping over the rows of $\mathbf{e}$ and updating the former according to Equation 5. Algorithms for obtaining $\mathbf{\Gamma}$ and $\mathbf{e}$ are given in (George-Williams & Patelli 2016).

3.2 *System Flow Equations*

Let X_{ij} be the magnitude of flow in edge e_{ij}, $\mho_i^+$; the set of nodes connected to the inlet of node *i*, $\mho_i^-$; the set of nodes connected to its outlet, $c_x^{\{i\}}$; its current capacity and x; its current state. The total inflow for source nodes (s) is zero and for sink nodes (t), the total outflow is zero which means $i \in s$ if $\mho_i^+ = \varnothing$ and $i \in t$ if $\mho_i^- = \varnothing$. Determining the performance level of output nodes (t), requires deriving the values of $X_{ij} \forall (i,j) \in \mathbf{e}$ for a given system state vector.

Assumption 1 of Section 2.1 is mathematically expressed by Equations 6, 7 and 8.

$$\sum_{j\in\mho_i^+, i\in s'} X_{ji}\alpha_{ji} \leq c_x^{\{i\}} \mid (i,j) \in \mathbf{e}, \quad \mho_i^+ \subset \mathbf{V} \quad (6)$$

$$\sum_{j\in\mho_i^-, i\in s} X_{ij} \leq c_x^{\{i\}} \mid (i,j) \in \mathbf{e}, \quad \mho_i^- \subset \mathbf{V} \quad (7)$$

Since there can't be negative flow in an edge, X_{ij} is such that $0 \leq X_{ij} \leq \Omega_{ij}$, where Ω_{ij} is the maximum

flow through the edge. If **lb** holds the lower bounds and **ub**, the upper bounds of flow through edges, then,

$$\mathbf{lb} = \{0\}_{k\times 1}, \quad \mathbf{ub} = \{\Omega_{ij}\}_{k\times 1}$$
$$\Omega_{ij} = \min\{c_{max}^{\{i\}}, c_{max}^{\{j\}}, l_{ij}\} \quad \forall (i,j) \in \mathbf{e} \tag{8}$$

where $c_{max}^{\{i\}}$ and $c_{max}^{\{j\}}$ are respectively the maximum capacities of nodes i and j. Applying Equations 6 and 7 across the system produces Equation 9,

$$\mathbf{\Theta}\{X_{ij}\}_{k\times 1} \leq \{c_x^{\{i\}}\}_{M\times 1} \mid (i,j) \in \mathbf{e}, \quad \forall i \in \mathbf{V} \tag{9}$$

where $c_x^{\{i\}}$ is the current capacity of node i and $\mathbf{\Theta}$ is related to $\mathbf{\Gamma}$ of the system as follows,

$$\mathbf{\Theta} = \{\theta_{iq}\}_{M\times k} \mid \theta_{iq} = \begin{cases} \gamma_{iq}, & i \in s \\ -\gamma_{iq}, & \gamma_{iq} < 0 \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

The second assumption in Section 2.1 is expressed by Equation 11 and across the entire system by Equation 12. Where $\eth$ is the number of intermediate nodes and

$$\sum_{j\in\mho_i^-} X_{ij} - (1-\varepsilon_x^{\{i\}}) \sum_{j\in\mho_i^-} X_{ji}\alpha_{ji} = 0 \mid (i,j) \in \mathbf{e} \tag{11}$$

$$\mathbf{\Phi}\{X_{ij}\}_{k\times 1} = \{0\}_{\eth\times 1} \quad \forall (i,j) \in \mathbf{e} \tag{12}$$

$$\mathbf{\Phi} = \{\phi_{\lambda q}\}_{\eth\times k} \mid \phi_{\lambda q} = \begin{cases} (1-\varepsilon_x^{\{p\}})\gamma_{pq}, & \gamma_{pq} < 0 \\ \gamma_{pq}, & \text{otherwise} \end{cases}$$
$$\lambda = 1,2,...,\eth \mid \eth < M \quad f: \lambda \to p \quad \forall p \in (s \cup t)' \tag{13}$$

$\varepsilon_x^{\{i\}}$, the proportion of flow dissipated by node i in state x. $\mathbf{\Phi}$ and $\mathbf{\Gamma}$ are related by Equation 13, which suggests every row in $\mathbf{\Phi}$ corresponds to a row in $\mathbf{\Gamma}$ associated with an intermediate node. Equations 9 and 12 are the flow equations of the system and should be solved to determine $\{X_{ij}\}_{k\times 1}$. Given $\mathbf{\Phi}$, $\mathbf{\Gamma}$ and $\{c_x^{\{i\}}\}_{M\times 1}$, a linear programming algorithm can be employed to solve these equations.

The objective of the optimization is to maximise flow from sources. Therefore, the objective function, $\mathbf{\Psi}$ is given by the negative sum of flow through all edges emanating from sources as expressed by Equation 14.

$$\mathbf{\Psi} = -\sum_{j\in\mho_i^-}\sum_{i\in s} X_{ij}$$
$$= -\{\psi_q\}_{1\times k}\{X_{ij}\}_{k\times 1} \mid \psi_q = \sum_{i\in s}\gamma_{iq} \tag{14}$$
$$q = 1,2,...,k$$

3.2.1 *Accounting for Bidirectional Flows*

The equations derived so far are with the assumption that process flow through an edge is unidirectional. For systems like power transmission or water supply networks, process flow through an edge may be possible in both directions. Under these conditions, the derived flow equations still remain valid but require that the range of acceptable flows across an edge be redefined. Normally, a bidirectional or undirected edge is represented by reciprocal edges. Two edges are said to be reciprocal if they originate and terminate on the same pair of nodes but allow process flow in opposite directions. For instance, edges e_{ij} and e_{ji} are reciprocal and together represent a single edge. Though flow is possible in both directions, at a given instance process flows in only one direction. Using this fact, one of the edges, say e_{ij} is arbitrarily chosen as reference. The lower bound of its flow is extended to $-\Omega_{ij}$; in other words, $-\Omega_{ij} \leq X_{ij} \leq \Omega_{ij}$ and the upper bound for edge, e_{ji} set to 0 (i.e., $\Omega_{ji} = 0$ implying $X_{ji} = 0$). If the index of e_{ij} in **e** is y and that of e_{ji}; y', then, $\mathbf{lb}(y) = -\Omega_{ij}$ and $\mathbf{ub}(y') = 0$ (see Equation 8). Following flow calculation, a negative X_{ij} signifies flow is in the direction specified by edge e_{ji}, hence, $X_{ji} = |X_{ij}|$ and $X_{ij} = 0$.

3.3 *Deriving System Output*

lb, **ub**, $\mathbf{\Theta}$, $\mathbf{\Phi}$, $-\{\psi_q\}_{1\times k}$, $\{\varepsilon_x^{\{i\}}\}_{M\times 1}$ and $\{c_x^{\{i\}}\}_{M\times 1}$ are the parameters required by the interior-point algorithm to determine the set $\{X_{ij}\}_{k\times 1}$. Let $\boldsymbol{\beta} \mid \boldsymbol{\beta} = \{\mathbf{w}_j\}^z \in \mathbb{R}^{z\times M'}$ be the matrix of all possible combinations of node states such that each row, $\mathbf{w}_j \mid j = 1,2,...,z$ represents a system state vector. If $A \Cap B$ denotes the matrix of all possible combinations of members of sets A and B, then, $\boldsymbol{\beta} = \{1,2,...,n_1\} \Cap \{1,2,...,n_2\} \Cap ... \Cap \{1,2,...,n_{M'}\}$, where n_i is the number of states of node i, $z = \prod_{i=1}^{M'} n_i$ and M', the total number of nodes including external CCI. To determine system flow corresponding to a given $\mathbf{w}_j$, the values of $\{c_x^{\{i\}}\}_{M\times 1}$, $\{\varepsilon_x^{\{i\}}\}_{M\times 1}$ and $\mathbf{\Phi}$, such that $x \in \mathbf{w}_j$, are first deduced. If none of the system nodes is prone to flow losses, then, only $\{c_x^{\{i\}}\}_{M\times 1}$ needs to be calculated, the rest remain static $\forall i \in \{1,2,..,M\}$. When $\{X_{ij}\}_{k\times 1}$ is obtained, the set, $\boldsymbol{\eta} \mid \boldsymbol{\eta} \in \mathbb{R}^{M\times 1}$ of flow through the nodes of the system is given by $\mathbf{\Theta}_{M\times k}\{X_{ij}\}_{k\times 1}$.

4 THE ALGORITHMS

4.1 *Incorporating Component Reconfiguration*

It was established in Section 3 that some system components may be subjected to minimum load requirements, below which they are shut down. This affects the affective value of system output and should therefore be taken into consideration during its analysis.

Owing to flow redistribution, shutting down a component/branch may augment flow in other branches and nodes which were originally qualified for shut down may no longer be. For this reason, a recursive procedure is employed in determining the output of the system.

Let $\boldsymbol{\Lambda}_g \mid \boldsymbol{\Lambda}_g = \{\Lambda_i\}_{M\times 1}$ be the vector containing the minimum threshold, $\Lambda_i \mid i = 1, 2, ..., M$ of all system components, $\boldsymbol{f}_1 = \{c_x^{\{i\}}\}_{M\times 1}$, $\boldsymbol{f}_2 = \{\varepsilon_x^{\{i\}}\}_{M\times 1}$ and $\boldsymbol{\tau}$, the set containing all other parameters required for system flow calculation as listed in Section 3.3. If $\boldsymbol{\eta}_j$ is the vector of effective flow through system nodes corresponding to state vector $\mathbf{w}_j$, then, Algorithm 1 presents the recursive procedure for determining system flow. In this algorithm, the symbol $\odot$ denotes element-wise multiplication of two vectors. Nodes are shut down in descending order of their degree of inadequacy (i.e., absolute difference between current and threshold flows) and only operating nodes with non-zero thresholds are considered. Shutting down zero-threshold nodes does not make any difference on the current system flow.

Algorithm 1 Recursive procedure for system flow calculation given non-zero minimum load condition on at least one system component

Require: $\boldsymbol{\eta}_j$, $\boldsymbol{f}_1$, $\boldsymbol{f}_2$, $\boldsymbol{\Lambda}_g \mid \boldsymbol{\Lambda}_g^T\boldsymbol{\Lambda}_g > 0$, $\boldsymbol{\tau}$

1: **function** RECONFIGURE($\boldsymbol{\eta}_j$, $\boldsymbol{f}_1$, $\boldsymbol{f}_2$, $\boldsymbol{\Lambda}_g$, $\boldsymbol{\tau}$)
2: $[a, b] = min\left(\{\boldsymbol{f}_1 > 0\} \odot \{\boldsymbol{\eta}_j - \boldsymbol{\Lambda}_g\}\right)$
3: **if** $a \geq 0$ **then**
4: $Exit$
5: **end if**
6: $\boldsymbol{f}_1(b) \leftarrow 0$, $\boldsymbol{f}_2(b) \leftarrow 0$ ▷ Update vectors
7: $flag \leftarrow 0$ ▷ Set recursive indicator
8: **while** $flag \leftarrow 0$ **do**
9: $\boldsymbol{\eta}_j \leftarrow \boldsymbol{\Theta}_{M\times k}\{X_{ij}\}_{k\times 1}$ ▷ Calculate flow
10: *Call lines 2 to 6*
11: **end while**
12: **return** $\boldsymbol{\eta}_j$
13: **end function**

4.2 *CCF Modelling*

When a CCI is also a component of the system, the modelling procedure does not deviate from what has already been established in earlier sections. For CCI residing outside the system boundary, a little manipulation is desirable to keep the computational time low. If σ is the probability of CCF, then, the event is represented by a binary-state node for which state 1; with probability $1 - \sigma$ is arbitrarily chosen as the state when CCF does not occur and state 2; with probability σ, as the CCF triggering state. CCIs do not directly influence system flow, for which reason they are not included in its network model. To ensure this does not pose a hitch to the flow equations derived and their implementation, node numbering starts with nodes that directly influence flow and continue with external CCI. This implies, external CCI numbering starts from $M + 1$ and for systems with no external CCI, $M' = M$. Quantifying the effect of CCF on the system

Algorithm 2 Procedure for CCF evaluation

Require: $\boldsymbol{Xout}$, $\boldsymbol{Xout'}$, $\boldsymbol{Xpath}$, $\boldsymbol{D}_i \neq \emptyset$

1: **function** EVAL(i,$\boldsymbol{\beta}$,$\boldsymbol{Xout}$,$\boldsymbol{Xout'}$,$\boldsymbol{Xpath}$)
2: $tstates \leftarrow$Vector of affected nodes
3: $counter \leftarrow 1$ ▷ pre-set counter
4: **while** $counter \leq |tstates|$ **do**
5: $index \leftarrow \boldsymbol{D}_i(:, 1) == tstates(counter)$
6: $g_1 \leftarrow \boldsymbol{D}_i(index, 2)$, $g_2 \leftarrow \boldsymbol{D}_i(index, 3)$
7: $g_3 \leftarrow \boldsymbol{D}_i(index, 4)$
8: $\boldsymbol{\Upsilon} \leftarrow \{0\}_{z\times|g_1|}$
9: **for** $j \leftarrow 1$ to $|g_1|$ **do**
10: $v_1 \leftarrow \boldsymbol{\beta}(:, g_1(j)) == g_2(j)$
11: $v_2 \leftarrow \boldsymbol{\beta}(:, i) == g_1(j)$
12: $\boldsymbol{\Upsilon}(v_1 \cap v_2, j) \leftarrow 1$
13: **end for**
14: $index2 \leftarrow$rows of $\boldsymbol{\Upsilon}$ with at least a 1
15: **for** $l \leftarrow index2$ **do**
16: $index3 \leftarrow \boldsymbol{\Upsilon}(l, :) > 0$
17: $\mathbf{w}_l \leftarrow$get state vector l
18: $\mathbf{w}_l(g_1(index3)) \leftarrow g_3(g_1(index3))$
19: $l' \leftarrow$new index of $\mathbf{w}_l$ in $\boldsymbol{\beta}$
20: $\boldsymbol{Xout'}(end + 1, :) \leftarrow \boldsymbol{Xout}(l', :)$
21: **end for**
22: $\boldsymbol{Xpath}(end + 1 : end + l) \leftarrow index2$
23: $counter \leftarrow counter + 1$ ▷ advance
24: **end while**
25: **return** $\boldsymbol{Xout'}$, $\boldsymbol{Xpath}$
26: **end function**

entails looping over all CCGs in the system. For each CCG, all state vectors in $\boldsymbol{\beta}$ containing the triggering state of the CCI with at least one CCG member in its vulnerable state are identified. The designated node performance level changes are made and the corresponding system flow calculated. This procedure, for one CCG is described by Algorithm 2. Where, i is the CCI's node ID, $\boldsymbol{Xout} \in \mathbb{R}^{z\times|t|}$; the matrix of performance levels of output nodes, such that each row is matched to a row in $\boldsymbol{\beta}$ and each column to an element in t, $\boldsymbol{Xpath}$; the indices of state vectors susceptible to CCF and $\boldsymbol{Xout'}$; matrix of performance levels of output nodes corresponding to these indices after CCF.

4.3 *Overall System Analysis*

The key system analysis tasks are determining $\boldsymbol{Xout}$ and $\boldsymbol{Xprob}$; the vector of probabilities corresponding to $\boldsymbol{Xout}$. Each element of $\boldsymbol{Xprob}$ is the product of the combination of probabilities corresponding to the state vector, $\mathbf{w}_i \mid i = 1, 2, ..., z$. Owing to the possibility of external CCI, only the first M elements of

$\mathbf{w}_i$ are considered in deriving $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$. If $\mathbf{w}'_i$ is the resultant state vector, the output flow obtained is assigned to all rows, i, of $\boldsymbol{\beta}$ for which $\mathbf{w}'_i \subset \mathbf{w}_i$. Let π_{rj}

Algorithm 3 Procedure for overall system analysis

Require: Node and system data, $\boldsymbol{\beta}$

function ANALYSE($\mathbb{S}$, $\boldsymbol{\beta}$)
  initialise all data storage arrays
  $\mho \leftarrow \{1, 2, ..., z\}$
  $index \leftarrow$set of CCI node IDs
  while $\mho \neq \emptyset$ **do**
    $i \leftarrow \mho(1)$ ▷ get state vector index
    get $\mathbf{w}'_i$, $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$
    $\boldsymbol{\eta}_i \leftarrow$get node flows
    $\boldsymbol{\eta}_i \leftarrow$RECONFIGURE($\boldsymbol{\eta}_i$, $\boldsymbol{f}_1$, $\boldsymbol{f}_2$, $\boldsymbol{\Lambda}_g$, $\boldsymbol{\tau}$)
    $I \leftarrow$all i of $\boldsymbol{\beta} \mid \mathbf{w}'_i \subset \mathbf{w}_i$
    $\boldsymbol{Xout}(I, :) \leftarrow \boldsymbol{\eta}_i(t)$ ▷ store output flows
    update $\boldsymbol{Xprob}\ \forall \mathbf{w}_i \mid i \in I$
    $\mho(I) \leftarrow \emptyset$ ▷ delete indices from set
  end while
  for $j \leftarrow index$ **do**
    ($\boldsymbol{Xout}'$, $\boldsymbol{Xpath}$)$\leftarrow$EVAL(j,...,$\boldsymbol{Xpath}$)
  end for
  $\boldsymbol{Xout}(\boldsymbol{Xpath}, :) \leftarrow \boldsymbol{Xout}'$
  for $r \leftarrow 1$ to $|t|$ **do**
    get π_{r1}, π_{r2},...,π_{rN}
    for $j \leftarrow 1$ to N **do**
      $h_{rj} \leftarrow$rows where $\boldsymbol{Xout}(:, r) = \pi_{rj}$
      $\mu_{rj} \leftarrow \sum \boldsymbol{Xprob}(h_{rj})$
    end for
  end for
  return Π, $\Bbbk$, H
end function

be the j^{th} performance level of output node r; r being its position in t, μ_{rj} the probability of this performance level and h_{rj} the set of state vector indices giving rise to π_{rj}. If Π, $\Bbbk$ and H are arrays respectively holding sets of π_{rj}, μ_{rj} and h_{rj} for all output nodes, then the overall system analysis entails their determination, as illustrated by Algorithm 3. They, together with $\boldsymbol{\beta}$ and t, make up the performance signature of the system.

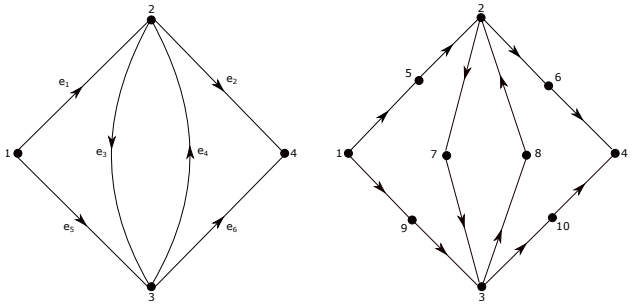
4.3.1 *Reliability and Performance Indices*

From the output of Algorithm 3, the system's reliability and performance indices can be deduced. Only two of these (availability and expected performance) are considered here, readers are referred to (Lisnianski, Frenkel, & Ding 2010) for a list of indices pertinent to MSS performance evaluation. The availability ($A_v^{\{r\}}$) of an MSS output node r is the probability that its performance level is at least v units while its expected performance ($g^{\{r\}}$) refers to its mean performance within a specified period. These are defined thus;

$$A_v^{\{r\}} = \sum_{j \forall \pi_{rj} \geq v} \mu_{rj} \qquad g^{\{r\}} = \sum_j \mu_{rj}\pi_{rj} \tag{15}$$

5 CASE STUDIES

5.1 *Example 1: A multi-state bridge system*



(a) Network model (b) Modified network

Figure 1: A multi-state bridge system with unreliable edges

Shown in Figure 1a is the network model of a 4-node, 6-edge bridge system. It's taken from (Yeh 2015), where the aim is to determine the probability that at least 3 units of flow are transmitted from node 1 to 4 using an improved sum-of-disjoint-product technique.

5.1.1 *Solution Procedure*

Owing to assumption 1 of Section 2.1, the unreliable edges e_1 to e_6 are respectively represented by nodes 5 to 10 in the modified network model shown in Figure 1b. Since there is no CCI, $M' = M = 10$, $z = 288$ and $\mathbf{D}_i = \emptyset \forall i \in \{1, 2, ..., 10\}$, the sets s and t are respectively $\{1\}$ and $\{4\}$. Nodes 1 to 4 are assigned a constant capacity of 4 units (since they are perfectly reliable), determined by the cumulative maximum allowable flow across e_1 and e_5. State assignment of each of the other nodes proceeds in ascending order of capacity. The capacity of each edge of the modified network is assumed to be infinite and since there are no threshold flow restrictions on nodes, $\Lambda_g = \{0\}^{10}$.

5.1.2 *Analysis Outcome*

The analysis took 6.922 seconds on a 1895.257MHz AMD Opteron (tm) 6168 processor and yielded the following outcome;

$$\Bbbk = \{0.0111, 0.1058, 0.2717, 0.4073, 0.2041\}$$

$$A_3^{\{1\}} = 0.4073 + 0.2041 = 0.6114 \tag{16}$$

$$\Pi = \{0, 1, 2, 3, 4\}, \quad g^{\{1\}} = 2.6875$$

Two additional scenarios were considered; the first assuming a 10% flow loss through e_1 and e_6 when they

respectively exist in states 3 and 2. The second assumes in addition, a minimum threshold load condition of 2 imposed on node 2 (i.e., $\Lambda_2 = 2$). The first results in 19 different performance levels with $A_3^{\{1\}}$ and $g^{\{1\}}$ respectively obtained as 0.3515 and 2.6251. It required the same computation time as the original case and the lower availability signifies the system exists more in the lower output levels, consequent of the load losing attributes of e_1 and e_6. The second scenario yields $A_3^{\{1\}} = 0.3515$, $g^{\{1\}} = 2.0884$, 10 distinct performance levels and took 12.304 seconds. The constant availability suggests the additional condition filters off only the lower performance levels. This is confirmed by the fewer performance levels of the system and its lower expected output. The overshoot in computation time is due to the need to recursively calculate system flow as a result of the minimum load condition imposed on node 2. In total, 540 calls were made to the flow calculation algorithm compared to 288 in the other scenarios.

5.2 *Example 2: Series-Parallel System with CCF*

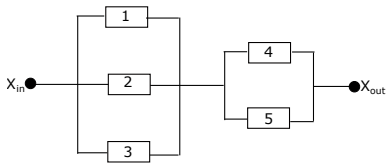


Figure 2: Series-parallel system with CCG

Table 1: Reliability and performance data of components

Component	Availability	Nominal Capacity
1	0.90	1.00
2	0.80	2.00
3	0.72	2.00
4	0.90	2.00
5	0.80	3.00

Figure 2 shows the block diagram of a simple series-parallel system with flow from X_{in} to X_{out}. It consists of two CCG; components 1 and 2 belong to CCG-1 with probability $\sigma_1 = 0.1$ and components 4 and 5; to CCG-2 with probability $\sigma_2 = 0.2$. It was initially presented as Example 4.15 in (Levitin 2005), where the UGF technique was used to derive its performance distribution and availability relative to a flow of 2 units. All components are binary-state and their properties are as given in Table 1.

5.2.1 *Solution Procedure*

Shown in Figure 3 is the network model for the system in which $s = \{1, 2, 3\}$ and $t = \{6\}$. Node 6 is assigned a constant capacity of 5 units, derived from the sum of the maximum flows through nodes 1, 2 and 3. Node X_{in} has been discarded since its removal doesn't affect flow across the system. For this system,

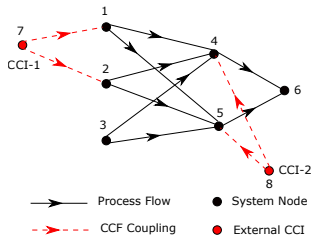


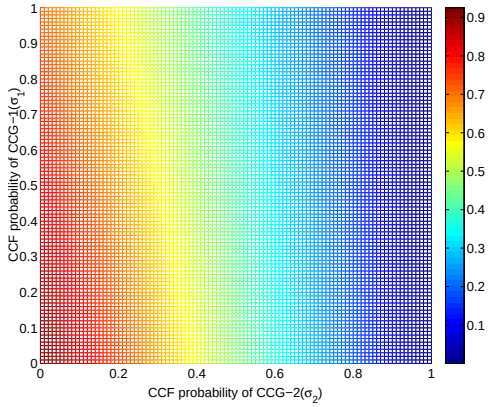
Figure 3: Network model for series-parallel system in Figure 2

$M = 6$, $M' = 8$, $z = 128$, $\Lambda_g = \{0\}^6$ and the CCG properties of nodes 7 and 8 are;

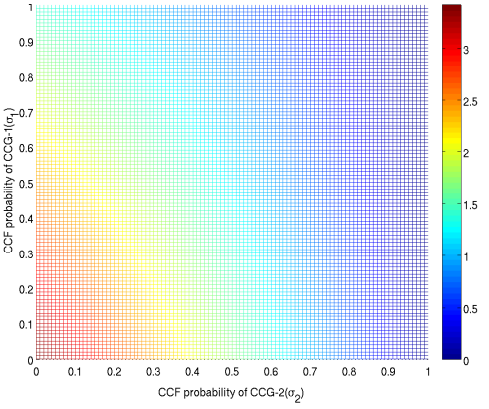
$$\mathbf{D}_7 = \begin{pmatrix} 2 & 1 & 1 & 2 \\ 2 & 2 & 1 & 2 \end{pmatrix}, \quad \mathbf{D}_8 = \begin{pmatrix} 2 & 4 & 1 & 2 \\ 2 & 5 & 1 & 2 \end{pmatrix} \tag{17}$$

The links are assumed to have an infinite capacity.

5.2.2 *Analysis Outcome*



(a) Availability ($A_3^{\{1\}}$)



(b) Mean Output ($g^{\{1\}}$)

Figure 4: Sensitivity of system performance indices to σ_1 and σ_2

$$\mathbb{k} = \{0.2419, 0.0356, 0.2, 0.2239, 0.0299, 0.2687\}$$

$$A_2^{\{1\}} = 0.2 + 0.2239 + 0.0299 + 0.2687 = 0.7225$$

$$\Pi = \{0, 1, 2, 3, 4, 5\}, \quad g^{\{1\}} = 2.5705 \tag{18}$$

The outcome presented in Equation 18 was obtained in 0.919 seconds on the same computer used in Example 1. Sensitivity of system performance to variations in CCF probability (σ_1 and σ_2) was also investigated. Using a probability interval of 0.01 for each of σ_1 and σ_2, the procedure would require $0.919\left(\frac{1+0.01}{0.01}\right)^2 = 9374.719$ seconds if a complete system analysis was run for each probability combination. However, using the performance signature of the system, only 96.205 seconds were required and the results are presented in Figure 4. From the relative colour change along the two axes, it's clear system performance is more sensitive to σ_2. This is not surprising, as the failure of CCG-2 implies the complete failure of the system. Therefore, σ_2 should be prioritized when reducing σ_1 and σ_2 under economic constraints. To show the applicability of the methodology to systems with multiple competing demand, a second demand point is introduced between the two parallel blocks of Figure 2. The effective demand on the system is equally shared between the two sinks such that each exerts a constant demand of 2.5 units. The system's new network model is shown in Figure 5 and $t = \{6, 7\}$. The performance indices of the output nodes are respectively derived as, $A_2^{\{1\}} = 0.457$, $g^{\{1\}} = 1.5931$, $A_2^{\{2\}} = 0.6731$ and $g^{\{2\}} = 2.1206$. One can clearly see that node 7 (output 2) sinks more flow than node 6 even though they have equal demand. This can be attributed to its location relative to the sources and CCG-2. Each time CCG-2 fails, the entire system flow is redirected to node 7.

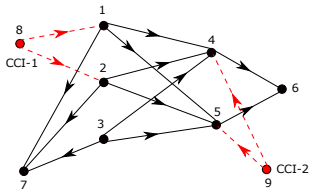


Figure 5: Network model for system with 2 output nodes

5.3 *Comments*

The case studies were analysed in short times and they yielded outcomes that agree with previous results. However, Algorithm 3 suggests the computation time is influenced by z. Therefore, large systems require a huge computational effort. Work is under way to derive an expression for the time complexity of the approach and investigate its efficiency relative to the UGF technique. It will also be extended to partial CCF, nested CCG and its application to time-dependent systems illustrated.

6 CONCLUSIONS

The case studies presented have illustrated the applicability of the approach to systems of complex structure and those susceptible to operational dynamics like CCF, minimum load restrictions, partial flow losses, and competing demand. The last three attributes are also limitations of the well-known UGF and sum-of-disjoint-product techniques. The performance signature concept introduced has been proven to bring tremendous gains in computation time. It comes in handy in non-structural optimization problems, epistemic uncertainty propagation, sensitivity analysis and other problems where only state probabilities of nodes are varied. Finally, the use of matrices to define the structure and flow across the system makes the approach intuitive enough for any system architecture and easily programmable on a computer. It, therefore, is an efficient and dependable computational tool, applicable to realistic multi-state systems.

ACKNOWLEDGMENTS

The authors would like to acknowledge the gracious support of this work through the EPSRC and ESRC Centre for Doctoral Training on Quantification and Management of Risk & Uncertainty in Complex Systems & Environments.

REFERENCES

George-Williams, H. & E. Patelli (2016). A hybrid load flow and event driven simulation approach to multi-state system reliability evaluation. *Reliability Engineering & System Safety 152*, 351 – 367.

Kojima, M., S. Mizuno, & A. Yoshise (1989). A primal-dual interior point algorithm for linear programming. In N. Megiddo (Ed.), *Progress in Mathematical Programming*, pp. 29–47. Springer New York.

Levitin, G. (2004). A universal generating function approach for the analysis of multi-state systems with dependent elements. *Reliability Engineering & System Safety 84*(3), 285 – 292.

Levitin, G. (2005). *The Universal Generating Function in Reliability Analysis and Optimization*. Springer-Verlag London Limited.

Lin, Y.-K. (2002). Using minimal cuts to evaluate the system reliability of a stochastic-flow network with failures at nodes and arcs. *Reliability Engineering & System Safety 75*(1), 41 – 46.

Lisnianski, A., I. Frenkel, & Y. Ding (2010). *Multi-State System Reliability Analysis and Optimization for Engineers and Industrial Managers*. Springer-Verlag London Limited.

Mehrotra, S. (1992). On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization 2*(4), 575–601.

Yeh, W.-C. (2008). A simple minimal path method for estimating the weighted multi-commodity multistate unreliable networks reliability. *Reliability Engineering & System Safety 93*(1), 125 – 136.

Yeh, W.-C. (2015, Dec). An improved sum-of-disjoint-products technique for symbolic multi-state flow network reliability. *Reliability, IEEE Transactions on 64*(4), 1185–1193.

Zang, X., D. Wang, H. Sun, & K. Trivedi (2003, Dec). A bdd-based algorithm for analysis of multistate systems with multistate components. *Computers, IEEE Transactions on 52*(12), 1608–1618.